Volume 6, Number 7 • July 2000
www.elementkjournals.com/sun
US $11.00

Inside Solaris™
Tips & Techniques for users of Sun Solaris

Accelerating development with Enterprise Java Beans

by Paul A. Watters

Many developers associate Java beans with cute, reusable, applet-based components that you use primarily to improve the user interface of Java applications. For example, you could design and implement a color picker component to dynamically modify color schemes for an applet based on current user preferences. However, as part of the Java 2 Enterprise Edition (J2EE) specification from Sun, there has emerged a new force in the server-side component market: Enterprise Java Beans (EJBs). J2EE provides a powerful framework for developing large-scale distributed applications. Another benefit of this framework is its ability to separate the presentation and business logic as shown in **Figure A**.

EJBs are used to encapsulate business logic and business rules into discrete components, which you can reuse across many different kinds of applications. Like client-side Java Beans, you can customize EJBs to suit a specific application. Each discrete component in a large system, which implements a discrete aspect of business logic, can be realized as an EJB, and EJBs are often created by Java developers to solve common business problems.

For example, many applications require that you create and store customer records in a remote database, and perform common operations on these records. Such operations can include inserting new data, updating existing records, and deleting outdated records. You can create EJBs that separate the common business logic behind these kinds of applications and provide an interface for applications to use when accessing objects of a particular abstract type.

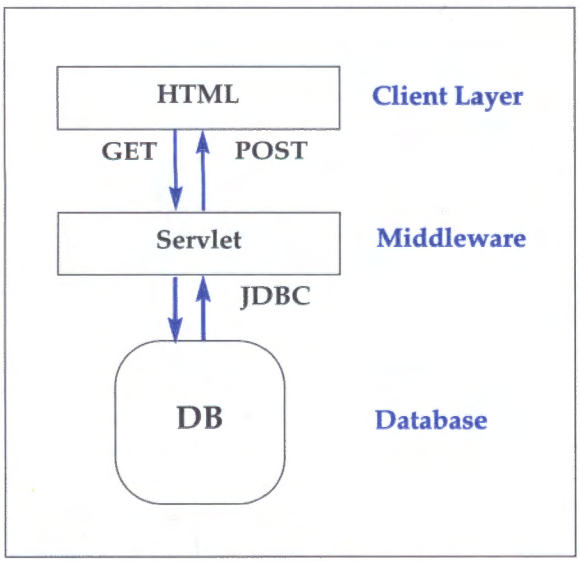

Figure A: *There's no separation of presentation or business logic in this scenario.*

Why EJBs?

Why is this separation necessary? It's always good design practice,

In this issue

elementk JOURNALS
A Division of Element K Press

especially in large development projects, to clearly separate application elements that are specific to a particular application from those that are common to many different kinds of applications. This promotes software reuse, particularly within an organization, and reduces the time to market for many applications. In addition, there's an emerging market for reusable software components that could potentially be a profitable spin-off for developers of an in-house project. This can help to reduce the cost of training developers to use EJBs in the first place.

To look at how EJBs can be used in a business context, let's imagine a situation where a client requires that you implement an online, catalog-based ordering system. An HTML client is generated dynamically by a Java Servlet, which then processes the data obtained through `GET` or `POST` parameters and organizes database operations around these parameters. For example, a customer might need to search for a specific product, so the HTML search page is generated by the servlet. The user then enters the search term into a search box and clicks the Search button.

Submitting data through POST passes parameters to the servlet, which then constructs a `SQL SELECT` statement, and passes this through to a database by using JDBC. The database processes the request, and passes the result back to the servlet, which then generates a results page for the search (hopefully with the desired item identified successfully). This process is shown in **Figure A**.

What this application lacks is a separation between the two most significant functions on the server-side: the presentation component, which is responsible for generating the HTML and/or JavaScript on the client side, and the business logic component, which implements the necessary classes to create, read, write, and update user data in the database. This means that every time the look and feel of the client front-end changes, the classes for the back-end, server-side functions may also be affected, if they aren't properly encapsulated.

Since encapsulation is one of the objectives of object-oriented programming, for very good reasons, we can separate presentation logic from business logic using EJBs. In this case, a servlet is an appropriate choice for taking care of the presentation layer, and you can deploy EJBs to process business logic. Thus, if the look and feel of a site changes significantly, the back-end functionality isn't affected. This also makes it easier, in multi-developer projects, to carve up development efforts along fairly specific tasks and responsibilities. **Figure B** shows the architecture using EJBs.

In addition, since distributed applications are much more complex than those that run on a single server, it's even more important to logically separate different functions. For example, there may be multiple databases or multiple application servers that use round-robin DNS or a similar scheme to efficiently process high-volume, Web-based transactions. In this case, a presentation layer may be served from one or more physically distinct machines.

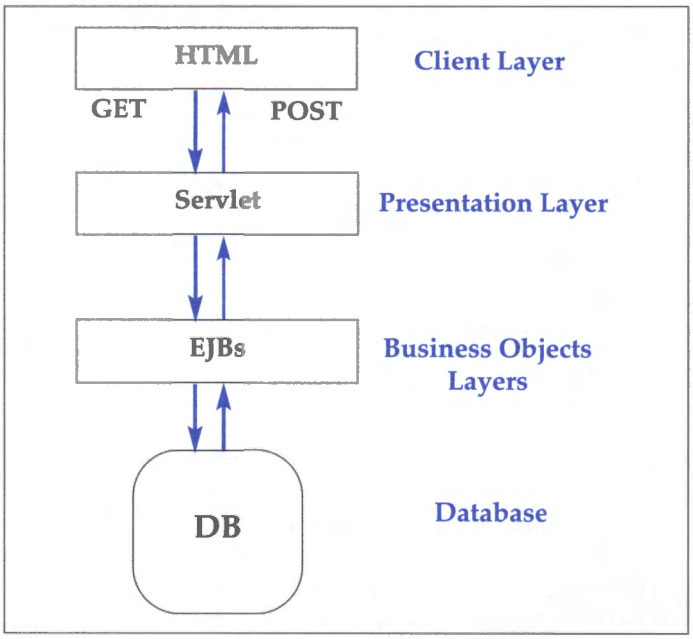

Figure B: *There's a complete separation of presentation and business logic in this scenario.*

Roles and responsibilities

Another benefit of separating responsibilities at the code level is that the human resources used in developing and deploying an EJB-based system can be logically assigned to the appropriate experts. The EJB specification from Sun suggests using several individual roles, which are important in determining the chain of responsibility in an operational context. For example, the EJB provider is a role associated with a business expert or analyst who has determined the optimal flow of information within an organization. However, there's also a second domain expert role known as the application assembler, who integrates the various EJB components available to satisfy the requirements of the application.

There are also two roles that fall under the systems umbrella. The first is the deployer role, where the individual is responsible for actually installing and configuring an EJB-based application. This requires expertise in setting up directory services and integration with existing software products. The second role is for the system administrator proper, whose role is to ensure that

the software works as specified and provide day-to-day monitoring and support. Although the system administrator in this role doesn't need to be a specialist in Java programming, he will need to be aware of several related issues:

- Setting up a Java Virtual Machine (JVM) and associated environment variables (for example, CLASSPATH)
- Archiving class files using the Java Archive (jar) command
- Understanding how Web servers and servlet runners are related
- Determining an optimal method for monitoring the status of live applications

Clearly, there's some overlap between the system administrator and deployer roles; although the deployer might actually copy the class files to a server, the system administrator needs to notify the deployer of the appropriate directory.

Choosing an EJB server

There are many application servers currently on the market that support EJBs and related networked object management technologies, such as CORBA. The main criteria for selecting an application server for EJBs include:

- Support for session and entity beans
- Support for persistence managed by the container
- Provision of an EJB deployment utility, which is preferably integrated with the development environment
- Support for standard SQL database commands using the JDBC interface

Several commercial application servers that meet these requirements are currently available, including BEA Weblogic (www.beasys.com) and ObjectSpace's Voyager (www.objectspace.com). However, these aren't readily integrated into an Integrated Development Environment (IDE).

The only offering that currently provides this level of integration comes from Inprise, whose Application Server (IAS) supports EJBs and CORBA, and is fully compliant with the Java 2 Enterprise Edition specification. JBuilder and IAS are by far the best commercial solution for EJB development on Solaris.

IAS is fully integrated with the JBuilder development environment that's available for Solaris, meaning that a single developer can also take responsibility for deployment roles, especially during testing. There's also a free community version of JBuilder for Solaris, which is available from www.inprise.com.

J2EE reference implementation

If you're not ready to invest in commercial software for this emerging technology, then Sun provides a reference implementation for J2EE, including EJB support, which is available as a free download from java.sun.com. However, potential users should be aware that this server isn't optimized for production use; it's perfectly adequate, however, for testing and development. Currently, the J2EE reference implementation is only available for Solaris SPARC and not Solaris Intel.

Once you have proceeded through registration, download the file j2sdkee1_2-solsparc. sh, which is a shell archive of approximately 10 MB. Change the permissions on the file to executable, and run the installation program by typing

```
unix% chmod +x j2sdkee1_2-solsparc.sh
unix% ./j2sdkee1_2-solsparc.sh
```

or by typing

```
unix% sh ./j2sdkee1_2-solsparc.sh
```

Next, you're asked (again!) to agree to the FCS Release Binary Software Evaluation Agreement. After scrolling through the license by pressing the [spacebar], you're asked:

```
Do you agree to the above license terms? [yes or no]
```

Type *yes* for the installation to proceed. You'll see a series of messages similar to the following displayed on your screen:

```
Unpacking...
Checksumming...
0
0
Extracting...
Archive:  ./install.sfx.18308
   creating: j2sdkee1.2/
   creating: j2sdkee1.2/lib/
   creating: j2sdkee1.2/lib/classes/
   creating: j2sdkee1.2/lib/security/
   inflating: j2sdkee1.2/lib/security/server.policy
   inflating: j2sdkee1.2/lib/security/client.policy
   creating: j2sdkee1.2/lib/dtds/
   inflating: j2sdkee1.2/lib/dtds/application-
   ➥client_1_2.dtd
   inflating: j2sdkee1.2/lib/dtds/application_1_2.dtd
   inflating: j2sdkee1.2/lib/dtds/ejb-jar_1_1.dtd
   inflating: j2sdkee1.2/lib/dtds/web-app_2_2.dtd
...
Done.
```

After the files have been extracted, you need to move the distribution to your deployment directory (usually /usr/local, but possibly /opt as well). You can do this by using the following command:

```
unix% mv j2sdkee1.2 /usr/local/
```

Next, you need to edit the user configuration script, which sets the CLASSPATH and associated properties (such as JDBC driver name). The userconfig.sh file should be contained in the /usr/local/j2sdkee1.2/bin directory, if you installed to /usr/local. A typical userconfig.sh looks like:

```
J2EE_CLASSPATH=$CLASSPATH:
➥/usr/local/somejdbcdriver/jdbc.jar
export J2EE_CLASSPATH
JAVA_HOME=/opt/jdk1.2.2/solaris
export JAVA_HOME
```

Here, we extend the existing definition of the CLASSPATH to include a third-party JDBC driver contained in a Java Archive file, and set the JAVA_HOME variable for the Java 2 Standard Edition installation (this must be installed prior to running J2EE).

Your next step is to read the "Configuration Guide," which is supplied with the J2EE documentation (downloaded separately), in order to enable each of the components, including the EJB server. For example, you'll need to set an `http.port` property for the EJB server (9191 by default), and a hostname and port number for the local CORBA object request broker (ORB), which is by default the localhost on port 1050. The Web server port is usually 8080, while SSL is supported on port 7070.

You can manually edit all of these configuration options in the /usr/local/j2sdkee1.2/config directory. Prior to bringing the J2EE services online, you should determine whether or not these services conflict with your existing services. ✳

Further reading

An excellent textbook on EJBs is *Enterprise JavaBeans* by Richard Monson-Haefel, which is now in its second edition (published by O'Reilly). You'll find some first-rate examples from the book at

www.oreilly.com/catalog/entjbeans2/chapter/ch04.html

There's also a superb comparison on a feature-by-feature basis of all currently existing EJB servers at

www.flashline.com/Components/appservermatrix.jsp?sid=942353844093-3221821718-151

Sun and Palm team up for remote wireless access

by Clayton E. Crooks II

Sun Microsystems and iPlanet have announced a deal with palmtop computer manufacturer, Palm Computing, in an effort to greatly improve mobile access to corporate data. They have plans to develop an all-inclusive solution that will provide global wireless access to enterprise applications and services via Palm OS-based handheld computers.

The Palm Personal Digital Assistant (PDA) will use Sun's Star Office portal and iPlanet (a Sun/Netscape alliance) services to give customers wireless access to corporate applications and data. Sun has also licensed Palm's HotSync technology and will develop Java applets to let Solaris users synch data automatically between their desktops and personal digital assistants.

The iPlanet and Star Office solution will help enable mobile professionals to access their corporate applications and data wirelessly with their Palm VII Personal Digital Assistant. As part of the proposed solution, iPlanet plans to add support for Palm's Web clipping architecture to its new iPlanet Wireless Server software, enabling corporate users to access their enterprise data through the Palm.Net wireless service on a worldwide basis.

Easily access data

Simple wireless access to corporate data will ultimately change the way a mobile workforce accomplishes their work. As an example, sales representatives could have real-time pricing, customer information, and product availability at the click of a button. They could complete orders and make changes to inventory once the order is made. The wireless solution also could streamline other processes, such as manufacturing and financial processes. The combination of Palm.Net services with Sun and iPlanet solutions will allow for easy access to additional applications such as email and group calendars, all wirelessly from Palm OS products.

It's unclear how this solution will evolve. Palm Computing, Sun, and iPlanet plan to market the wireless solution to Internet and application service providers. Upon deployment of the solution, Palm intends to market the Palm.Net service as iPlanet-ready. They are also evaluating the SunTone Certification and Branding program for the new wireless services to be offered by the Palm.Net Web site.

Certification under the SunTone program would make Palm eligible to participate in cooperative marketing, financing, and consulting. This would be a tremendous benefit to Palm because these programs are supported by several thousands of Sun's sales, service, and marketing personnel worldwide. Additionally, iPlanet intends to market its applications as Palm-ready.

Desktop communications

Sun will develop software to connect desktop applications used on UltraSparc workstations to the Palm Pilot line-of-information organizers. Applications included with the Sun Solaris operating system, such as Desktop Mail and Calendar Manager, will be able to exchange data with Palm apps such as Address Book, Date Book, Memo Pad, and Palm Mail. By doing so, Sun is addressing the needs of UNIX system administrators and engineers in providing access to data secured on the popular Palm PDAs. Palm Pilots are the fastest-selling retail PDAs and have sold more than 400,000 units in the past two years, according to market research firm, PC Data.

Currently, there are groups of mostly unsupported applications for UNIX users. They rely on these collections of independently written freeware utilities for their data synchronization. UNIX users will benefit from having fully supported, cross-platform applications.

Sun will post the software for the Sun Ultra workstations to their Web site later this year and may develop full Java applications that would let Palm Pilots communicate with any Java-enabled device, such as Sun's Java Station. Sun is also considering integrating the software into the operating system, depending on the timing of the next Solaris update.

A wireless future?

Though wireless Internet services aren't yet widespread, companies are furiously working to secure a lead. America Online, for example, has announced multiple deals and programs to spread its popular instant messenger software to cell phones. AOL and Sun, as well, have separately been working on development projects.

Palm is also looking to tap other potential markets. With a staggering 70 percent market share in hand-held computers, Palm is focusing their future growth on corporate customers. The company plans to leverage its user base of executives and mobile professionals.

Further, Palm is aggressively expanding its wireless offerings, mainly through several licensing efforts. The Palm VII wireless device and accompanying Palm.Net service is seen as a niche product. In the future, they'll primarily focus on partnerships with handset manufacturers and other device makers.

Already, the company has struck deals to put its software on phones from Nokia and Motorola, and Sony is expected to use Palm software on its upcoming multimedia device. But Palm faces competition from Microsoft and the Symbian alliance, which are also developing software that will run Internet-enabled cell phones.

The mobile market is still very much open, with companies scrambling to create or keep their market share. The stakes for this market are especially large, as recent predictions by Lehman Brothers estimates that by 2007, half of all cell phones sold worldwide will be of the "smart" variety. It remains to be seen whose software will be on these phones, although the Sun and Palm alliance will undoubtedly play a part in determining the winner. ✳

Samba password encryption

by Don Kuenz

Enabling Samba's encrypted password functionality provides at least two benefits. First, encryption makes it harder for hackers to discover your Samba passwords. Second, and perhaps most important, password encryption makes Samba function much more smoothly on Windows clients. In this article, we'll show you how to enable Samba's password encryption.

Many times, Samba displays a *\\hostname is not accessible* error after you initially install it and attempt to access your server from a recent version of Windows. A plain text/encrypted password mismatch between Windows and Solaris causes this error. Recent versions of Windows enable encryption by default, while Samba disables encryption and uses a plain text password by default. Enabling plain text passwords on your Windows host solves the problem.

Plain text passwords

All versions of Windows use a registry key to control encryption. By default, all recent versions of Windows enable encryption. It order to disable it, you must add a key that enables plain text passwords into the registry. Listing A shows the appropriate registry keys for various versions of Windows.

Samba uses a TCP protocol named SMB. TCP wraps SMB information into packets, and then sends them out over the wire. Early on, when you first negotiate a session, the SMB protocol requires clients to send a packet containing a user name and a password.

As the name suggests, you can read plain text passwords by simply looking at SMB packets as they travel over the wire. Let's take a look at what a plain text password looks like.

Solaris comes with a handy tool named snoop, which displays the contents of packets. Let's use the `snoop` command:

```
snoop -d iprb0 -S -x 0
```

This displays an SMB packet that contains a user name and password. Snoop uses the `-d` argument to specify the network device that you wish to monitor. You can obtain a list of your own network device(s) by issuing the following command:

```
ls /etc/hostname.*
```

The last node of the file(s) listed contains your network device name(s). The `-S` and `-x 0` arguments tell snoop how to display the packet. Listing B shows what snoop displays during an SMB negotiation.

Can you spot our password in Listing B? It's the word PASSW0RD that appears close to the bottom of the packet. You can also see the user name, DON, that appears after the password. Now a hacker could use the combination of DON and PASSW0RD to access our Samba shares.

Let's take a closer look at the packet displayed in Listing B. First, all SMB messages begin with the characters SMB followed by a message type. This SMB message contains a type of 0x7300, which is a `SESSION_SETUP_ANDX` message. On most networks, the `SESSION_SETUP_ANDX` always contains a password and a user name. Second, the SMB protocol converts all user names and passwords to upper-case letters. Our actual user name is *don*, and our actual password is *Passw0rd*. That makes things slightly tougher for a malicious hacker.

In addition to creating a security hole, plain text passwords also cause other problems with Windows clients. Plain text makes it hard to browse Samba hosts. It also forces Windows to ask for a user ID and a password when you first access Samba shares after a reboot. Password encryption takes care of both problems.

Listing A: *Registry keys that enable plain text passwords for various versions of Windows*

```
Windows 95:
[HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\VxD\VNETSUP]
"EnablePlainTextPassword"=dword:00000001

Windows 98:
[HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\VxD\VNETSUP]
"EnablePlainTextPassword"=dword:00000001

Windows NT4-SP3 and later:
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Rdr\Parameters]
"EnablePlainTextPassword"=dword:00000001

Windows 2000:
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\
➥LanmanWorkStation\Parameters]
"EnablePlainTextPassword"=dword:00000001
```

Password encryption

You should find it fairly easy to enable Samba password encryption. First, edit smb.conf and add a line to the `[global]` section that says

```
encrypt passwords = yes
```

Next, use `kill -HUP <pid>` and specify the Process ID `<pid>` for each `smbd` and `nmbd` processes. The `smbd` and `nmbd` processes control the Samba functionality on your Solaris host. A `kill` command with a `-HUP` argument causes the processes to reinitialize using the new smb.conf file.

Next, you need to delete all of the registry keys shown in Listing A from all of your Windows clients. Windows stores password information in *.pwl files, which you can find under the Windows root directory (typically C:\WINDOWS). If you run into problems, you might find it useful to delete those files. In that case, when a user logs on next, Windows prompts him for a password and a password confirmation—no harm done.

You must complete one last task on your Solaris host before encryption becomes fully operational. You need to create a smbpasswd file (different from the `smbpasswd` command) that contains hashes used by the encryption algorithm. By default, Samba installs this file in the /usr/local/samba/private/ directory. The Samba source package comes with a handy script named mksmbpasswd.sh that helps you to create an initial smbpasswd file. Invoke the following commands to initially create your file:

```
cat /etc/passwd | mksmbpasswd.sh > \
  /usr/local/samba/private/smbpasswd

chown -R root /usr/local/samba/private

chmod 500 /usr/local/samba/private

chmod 600 \
  /usr/local/samba/private/smbpasswd
```

We use `chown` and `chmod` to protect `smbpasswd` from prying eyes. Finally, after you build this file, you use the `smbpasswd` command (not the file) to create passwords for your users. As root, you can use a command similar to

```
smbpasswd <userid> <password>
```

to change other user's passwords. Everyone else can use this command to change their own passwords. Read the ENCRYPTION.TXT file that comes with Samba if your site contains a lot of users and you wish to make the migration easier. Encryption allows your Windows clients to behave much more smoothly. It causes Windows to actually remember Samba passwords without making you enter them at the start of each session. You can also browse Samba hosts that appear in the Network Neighborhood.

Let's use `snoop` again to take a look at an SMB packet that uses encryption. Listing C shows such a packet.

Can you still spot our password in Listing C? At this point, probably only a cryptographic

Listing B: *The snoop command displays a plain text password, in this case the word Passw0rd.*

```
snoop -d iprb0 -S -x 0

zeus.crc.wy -> apollo.crc.wy length:  205  TCP D=139 S=1030
     Ack=2007521580 Seq=16433569 Len=151 Win=8679

   0: 0004 ac45 d2c8 0004 ac45 d7a2 0800 4500    ...E.....E....E.
  16: 00bf 5502 4000 8006 c9d6 c02a ad88 c02a    ..U.@......*...*
  32: ad82 0406 008b 00fa c1a1 77a8 592c 5018    ..........w.Y,P.
  48: 21e7 c74e 0000 0000 0093 ff53 4d42 7300    !..N.......SMBs.
  64: 0000 0010 0000 0000 0000 0000 0000 0000    ................
  80: 0000 0000 4518 0100 8196 0d75 0075 0068    ....E......u.u.h
  96: 0b32 0000 0034 5300 0018 0000 0000 0000    .2...4S.........
 112: 0001 0000 0038 0050 4153 5357 3052 4400    .....8.PASSW0RD.
 128: 0000 0000 0044 4f4e 0000 0000 0000 0044    .....DON.......D
 144: 4f4e 0043 5243 0057 696e 646f 7773 2034    ON.CRC.Windows 4
 160: 2e30 0057 696e 646f 7773 2034 2e30 0004    .0.Windows 4.0..
 176: ff00 0000 0200 0100 1300 005c 5c41 504f    ...........\\APO
 192: 4c4c 4f5c 4950 4324 0049 5043 00           LLO\IPC$.IPC.
```

Listing C: *The snoop command displays an encrypted password, in this case the word Passw0rd.*

```
snoop -d iprb0 -S -x 0

zeus.crc.wy -> apollo.crc.wy length:  205  TCP D=139 S=1033
     Ack=2366050351 Seq=19257287 Len=151 Win=8671

   0: 0004 ac45 d2c8 0004 ac45 d7a2 0800 4500    ...E.....E....E.
  16: 00bf 8204 4000 8006 9cd4 c02a ad88 c02a    ....@......*...*
  32: ad82 0409 008b 0125 d7c7 8d07 102f 5018    .......%...../P.
  48: 21df d080 0000 0000 0093 ff53 4d42 7300    !..........SMBs.
  64: 0000 0010 0000 0000 0000 0000 0000 0000    ................
  80: 0000 0000 4518 0100 8167 0d75 0075 0068    ....E....g.u.u.h
  96: 0b32 0000 00c5 5300 0018 0000 0000 0000    .2....S.........
 112: 0001 0000 0038 0033 5d00 faba b2d4 a471    .....8.3]......q
 128: 082d 90a5 4d46 c4ba dde0 01fa c5b9 6d44    .-..MF........mD
 144: 4f4e 0043 5243 0057 696e 646f 7773 2034    ON.CRC.Windows 4
 160: 2e30 0057 696e 646f 7773 2034 2e30 0004    .0.Windows 4.0..
 176: ff00 0000 0200 0100 1300 005c 5c41 504f    ...........\\APO
 192: 4c4c 4f5c 4950 4324 0049 5043 00           LLO\IPC$.IPC.
```

expert could decode our password. We still need to mention one more topic.

Revised security assessment

Encryption makes it much harder for a hacker to compromise Samba's security. However, a typical Solaris host runs other TCP services in addition to Samba. Unfortunately, many of these other services contain well-known security holes that hackers can use to compromise your system. Let's take a look at a common daemon, which contains a well-known hole.

A lot of hosts run a telnetd daemon to allow users telnet access into the host. We can use `snoop` once more to display packets pertaining to a telnet login. Using `snoop -d iprb0 -S` provides the best output to examine telnet passwords. **Listing D** shows the output from such a command.

As you can see, `snoop` makes it quite easy to determine a user's password during a telnet login. **Listing D** also shows an interesting phenomenon caused by echoing characters back to the screen. Notice that the characters in the user ID (don) appear twice: once when the user types the character, and again when the host echoes the character back to the telnet client for display on the screen. On the other hand, the characters in the password appear only once, because, for the sake of security, the host doesn't echo password characters back to the telnet client. Ironically, this very security measure only makes password cracking that much easier.

Samba encryption serves to strengthen Samba. Unfortunately, system security is only as strong as the weakest link, and in this case, the weakest link is telnetd. To build a secure host, you must always examine every part of it.

Listing D: *Snoop output showing packets pertaining to a telnet login*

```
apollo.crc.wy -> eos.crc.wy    length:   61  TELNET R port=1167 login:
  eos.crc.wy -> apollo.crc.wy length:   60  TELNET C port=1167
  eos.crc.wy -> apollo.crc.wy length:   60  TELNET C port=1167 d
apollo.crc.wy -> eos.crc.wy    length:   55  TELNET R port=1167 d
  eos.crc.wy -> apollo.crc.wy length:   60  TELNET C port=1167
  eos.crc.wy -> apollo.crc.wy length:   60  TELNET C port=1167 o
apollo.crc.wy -> eos.crc.wy    length:   55  TELNET R port=1167 o
  eos.crc.wy -> apollo.crc.wy length:   60  TELNET C port=1167 n
apollo.crc.wy -> eos.crc.wy    length:   55  TELNET R port=1167 n
  eos.crc.wy -> apollo.crc.wy length:   60  TELNET C port=1167
  eos.crc.wy -> apollo.crc.wy length:   60  TELNET C port=1167
apollo.crc.wy -> eos.crc.wy    length:   56  TELNET R port=1167
  eos.crc.wy -> apollo.crc.wy length:   60  TELNET C port=1167
apollo.crc.wy -> eos.crc.wy    length:   64  TELNET R port=1167 Password:
  eos.crc.wy -> apollo.crc.wy length:   60  TELNET C port=1167
  eos.crc.wy -> apollo.crc.wy length:   60  TELNET C port=1167 P
apollo.crc.wy -> eos.crc.wy    length:   54  TELNET R port=1167
  eos.crc.wy -> apollo.crc.wy length:   60  TELNET C port=1167 a
apollo.crc.wy -> eos.crc.wy    length:   54  TELNET R port=1167
  eos.crc.wy -> apollo.crc.wy length:   60  TELNET C port=1167 s
apollo.crc.wy -> eos.crc.wy    length:   54  TELNET R port=1167
  eos.crc.wy -> apollo.crc.wy length:   60  TELNET C port=1167 s
apollo.crc.wy -> eos.crc.wy    length:   54  TELNET R port=1167
  eos.crc.wy -> apollo.crc.wy length:   60  TELNET C port=1167 w
apollo.crc.wy -> eos.crc.wy    length:   54  TELNET R port=1167
  eos.crc.wy -> apollo.crc.wy length:   60  TELNET C port=1167 0
apollo.crc.wy -> eos.crc.wy    length:   54  TELNET R port=1167
  eos.crc.wy -> apollo.crc.wy length:   60  TELNET C port=1167 r
apollo.crc.wy -> eos.crc.wy    length:   54  TELNET R port=1167
  eos.crc.wy -> apollo.crc.wy length:   60  TELNET C port=1167 d
apollo.crc.wy -> eos.crc.wy    length:   54  TELNET R port=1167
  eos.crc.wy -> apollo.crc.wy length:   60  TELNET C port=1167
apollo.crc.wy -> eos.crc.wy    length:   56  TELNET R port=1167
  eos.crc.wy -> apollo.crc.wy length:   60  TELNET C port=1167
apollo.crc.wy -> eos.crc.wy    length:  180  TELNET R port=1167
➥Last login: Wed Mar
```

Conclusion

In this article, we've shown you how to enable password encryption in Samba. Recent versions of Windows enable encryption by default, but Samba disables encryption and uses plain text passwords by default. To enable encryption, you simply add a line to your smb.conf file and create a smbpasswd file that contains hashes. Keep in mind that Samba encryption is only one part of a robust security solution. ✳

Network security with Kerberos

by Boris Loza

In Greek mythology, Kerberos (Cerberus in the Latin spelling) is the three-headed dog that guarded the entrance to Hades. In the computer world, *Kerberos* is a network authentication protocol. It's designed to provide strong cryptography so that a client can prove its identity to a server (and vice versa) across an insecure network connection. After a client and server have used Kerberos to prove their identities, they can also encrypt all of their communications.

Massachusetts Institute of Technology created Kerberos in the mid 1980s as part of the Athena project. Today Kerberos version 5 (Kerberos V5) is the latest implementation and is available as a product from many different vendors for various computing platforms. In this article, we'll show you how to use Kerberos V5 in the Solaris environment as a solution to your network security problems.

How Kerberos works

When you log on to a workstation running Kerberos, you provide your username and password. The workstation sends the Key Distribution Center (KDC) on the Kerberos Server a message consisting of your username and the current time encrypted with your password. The KDC looks up your username, determines your password, and attempts to decrypt the encrypted time. If the server can decrypt the current time, it then creates a ticket-granting ticket, encrypts it with your password, and sends it back to you. The user can then contact the KDC to obtain tickets for any network service within the Kerberos realm. The *Kerberos realm* is the domain of servers and users who are known to the KDC.

As you can see from this scenario, the user's password is never transmitted on the network in clear text. Kerberos will give you credentials only if you have an entry in the Kerberos server's database. This database includes a Kerberos principal (an identifying string, which is often just your user name) and your Kerberos password. Every Kerberos user must have an entry in this database.

There are two types of KDC: master and slave. Each KDC contains a copy of the Kerberos database. The master KDC contains the master copy of the database, which it propagates to the slave KDC(s) at regular intervals. All database changes (such as password changes) are made on the master KDC(s).

Obtaining and installing Kerberos

Although Kerberos is freely available, only US and Canadian citizens can legally download it from MIT. To do so, go to **web.mit.edu/network/kerberos-form.html** and fill out the request form. After downloading the latest release, unpack it with a tar. You'll get source, documentation, and digital signature files all in .tar.gz format. As root, gunzip and untar each of the above .tar.gz files and switch into the <where-you-install>/krb-5.1.1.1/src directory. To create the Makefile, run `configure` with appropriate options. For example,

```
./configure -with-cc=gcc --prefix=/usr/local/kerberos
```

These options tell `configure` to use the GNU gcc compiler and install all binaries and man pages under the /usr/local/kerberos directory. For many other options, check the *Kerberos V5 Installation Guide* that comes with the distribution.

After `configure` is finished, make all binaries by typing `make` and install the utilities by typing `make install`.

Configuring the primary KDC

The machine that you choose as your Kerberos server is the key to your network security. Therefore, it must be as secure as possible. Make sure that:

- The server is physically secure and all the latest OS patches have been applied.
- There are no user accounts on the machine except for the Kerberos administrator.
- All unnecessary network services are commented out in the /etc/inetd.conf file.
- You have one more physically and computationally secure machine to serve as alternative server (as a slave KDC), in case there's a hardware problem or a network outage.

For detailed instructions on how to secure a Solaris server, you can check "Securing Solaris: Step-by-Step." Ref. No. SG110A from SANS.

For the sake of simplicity, let's assume that you have the following configuration:

- Domain name—tree.com
- Primary KDC—birch.tree.com
- Slave KDC—pine.tree.com
- Kerberos Client—oak.tree.com

Configuring the master KDC

Suppose you've already installed Kerberos on birch.tree.com following the instructions from the section "Obtaining and installing Kerberos." The next step is to modify the configuration files krb5.conf shown in **Listing A**, on the next page, and kdc.conf shown in **Listing B**, also on the next page, that come with the distribution (<where-you-install>/krb5-1.1.1/src/config-files) to reflect the correct information, such as the hostnames and the realm name for your realm. The entries you need to modify are shown in color. Note that the realm names are case sensitive. Place these files into the appropriate directories. For the krb5.conf, it's the /etc directory. For the kdc.conf,

it's /usr/local/var/krb5kdc. Note that you have to manually create the /usr/local/var/krb5kdc directory.

For an explanation of each entry in the configuration files, consult the krb5.conf(5) and kdc.conf(5) man pages and the *Kerberos V5 System's Administrators Guide* (comes with a distribution). The convention is to make a realm the same as your domain name, in upper-case letters.

Now we're ready to create a Kerberos database and the stash file using the kdb5_util utility. The stash file is a local copy of the master key that resides in encrypted form on the Master KDC's local disk. Use this file to authenticate the KDC to itself automatically before starting the Kerberos daemons during the server boot. If the stash file (/usr/local/var/krb5kdc/.k5.TREE.COM in the kdc.conf file in **Listing B**)

Listing A: *Kerberos KDC configuration file /etc/krb5.conf*

```
[libdefaults]
    default_realm = TREE.COM
    default_tkt_enctypes = des-cbc-crc
    default_tgs_enctypes = des-cbc-crc


[realms]
TREE.COM = {
    kdc = birch.tree.com:88
    kdc = pine.tree.com:88
    admin_server = birch.tree.com:749
}

[domain_realm]
    .tree.com = TREE.COM

[kdc]
    profile = /usr/local/var/krb5kdc/kdc.conf

[logging]
    kdc = FILE:/var/log/kdc.log
    default = FILE:/var/log/kdc.log
```

Listing B: *Kerberos KDC configuration file usr/local/var/krb5kdc/kdc.conf*

```
[kdcdefaults]
    kdc_ports = 88

[realms]
TREE.COM = {
    database_name = /usr/local/var/krb5kdc/principal
    admin_keytab = FILE:/usr/local/var/krb5kdc/kadm5.keytab
    acl_file = /usr/local/var/krb5kdc/kadm5.acl
    key_stash_file = /usr/local/var/krb5kdc/.k5.TREE.COM
    kadmind_port = 749
    max_life = 10h 0m 0s
    max_renewable_life = 7d 0h 0m 0s
    master_key_type = des-cbc-crc
    supported_enctypes = des-cbc-crc:normal
}
```

were compromised, it would allow unrestricted access to the Kerberos database. Make sure that this file is readable only by root and it isn't a part of a backup:

```
# /usr/local/kerberos/sbin/kdb5_util create -s
```

This file prompts you for the master key for the Kerberos database. This key can be any string. Choose a key that's very difficult to guess.

Next you need to create an Access Control List (acl) file:

```
#cat /usr/local/var/krb5kdc/kadm5.acl
*/admin@TREE.COM *
```

This gives the principal */admin@TREE.COM permission to change all of the database permissions on any principal permissions.

The next step is to set up the database entry for at least one Kerberos administrator. Run `kadmin.local` and use `addprinc`, as shown in **Listing C**.

To get information about this user, you can use getprinc with kadmin.local. Type ? inside of kadmin.local to see a list of all available requests.

Create the keytab file on the server. Kadmind uses this to determine what access it should give to administrators. You need to create the keytab entries for the principals kadmin/admin and kadmin/changepw. (These principals are placed in the Kerberos database automatically when you create it.) Now, stay in kadmin.local and run:

```
kadmin.local:  ktadd -k /usr/local/var/krb5kdc/
➥kadm5.keytab kadmin/admin kadmin/changepw
```

Edit the /etc/services file to include the services shown in **Listing D**.

Start the master KDC and the Kerberos administration daemons by typing:

```
#/usr/local/kerberos/sbin/krb5kdc
#/usr/local/kerberos/sbin/kadmind
```

If you'd like to start these servers automatically when the Kerberos Server is rebooted, you have to create a startup file in the /etc/init.d directory and link it with the corresponding files in the /etc/rc3.d. Now we're ready to go to pine.tree.com (our second KDC) and proceed with a Slave KDC installation.

Configuring the slave KDC

Setting up a slave KDC is relatively simple. You need to install Kerberos on the slave KDC (pine.tree.com) and edit the configuration files krb5.conf and kdc.conf the same as for the primary KDC.

Both slave and master KDCs need host principals in the Kerberos database. You can run kadmin on pine.tree.com to administer the Kerberos database remotely by using the following:

```
#/usr/local/kerberos/sbin/kadmin --p admin/admin
kadmin: addprinc --randkey host/birch.tree.com
kadmin: addprinc --randkey host/pine.tree.com
```

Add the service principal to the server's keytab so it can provide the Kerberized service:

```
kadmin: ktadd host/birch.tree.com
kadmin: ktadd host/pine.tree.com
```

In the Kerberos database directory (/usr/local/var/krb5kdc) on both your master and slave, you need to create a file called *kpropd.acl* and place in it all of the host principals for your KDCs:

```
#cat /usr/local/var/krb5kdc/kpropd.acl
host/birch.tree.com
host/pine.tree.com
```

On the slave KDC (pine.tree.com), add an entry for kpropd in inetd.conf:

```
krb5_prop  stream  tcp  nowait root
➥/usr/local/kerberos/sbin/kpropd kpropd
```

On the master KDC (birch.tree.com), dump the database into a file using kdb5_util:

```
#/usr/krb5/sbin/kdb5_util dump
➥/usr/local/var/krb5kdc/slave_datatrans
```

Now you can run kprop on the master to propagate the database to the slave:

```
#/usr/local/kerberos/sbin/kprop -f
➥/usr/local/var/krb5kdc/slave_datatrans
➥pine.tree.com
```

It's a good idea to set up a crontab job to propagate the database at regular intervals.

The final step in configuring the slave KDC is to run the KDC daemon:

```
#/usr/local/kerberos/sbin/krb5kdc
```

You're all set. Now let's connect oak.tree.com as a client machine.

Connecting a client

The Kerberos client is a machine that offers kerberized services such as telnet, ftp, r* services (rlogin, rcp, rsh), or even NFS and NIS+ network facilities. These services are modified to work with Kerberos.

Listing C: *Setting up the Kerberos administrator*

```
# /usr/local/kerberos/sbin/kadmin.local
Authenticating as principal root/admin@TREE.COM with password.
kadmin.local:  addprinc admin/admin
WARNING: no policy specified for admin/admin@TREE.COM;
➥defaulting to no policy
Enter password for principal "admin/admin@TREE.COM":
Re-enter password for principal "admin/admin@TREE.COM":
Principal "admin/admin@TREE.COM" created.
```

Listing D: *Lines added to our /etc/services file for Kerberos*

```
kerberos        88/udp    kdc      # Kerberos V5 KDC
kerberos        88/tcp    kdc      # Kerberos V5 KDC
klogin          543/tcp             # Kerberos authenticated rlogin
kshell          544/tcp   krcmd    # and remote shell
kerberos-adm    749/tcp             # Kerberos V5 admin/chpwd
kerberos-adm    749/udp             # Kerberos V5 admin/chpwd
krb5_prop       754/tcp             # Kerberos V5 slave propagation
eklogin         2105/tcp            # Kerberos auth. and encrypted rlogin
```

First you have to install Kerberos on the client the same as you did on all KDCs. Edit the /etc/krb5.conf file. You must add the appropriate Kerberos services into the /etc/services file as described in the section "Configuring the master KDC." Comment out any lines for telnet and ftp services in the /etc/inet/inetd.conf file. You'll use the kerberized versions that come with the Kerberos distribution. Add the lines shown in **Listing E**, on the next page, into the /etc/inet/inetd.conf file onoak.tree.com.

Don't forget to kill the inetd process with the -HUP option. In order to allow oak.tree.com to be a Kerberos client, you have to create a /etc/krb5.keytab file. Inside the kadmin, add the host as a principal:

```
#/usr/local/kerberos/sbin/kadmin --p admin/admin
kadmin: addprinc  -randkey host/oak.tree.com
```

You'll see a message that the principal host/oak.tree.com is created. Now, add its keytab entry in the oak's /etc/krb5.keytab file:

```
kadmin: ktadd oak.tree.com
```

This process securely shares a secret key to be used for communication between oak and the KDC server. You need to repeat this process for every host in your realm.

The next step is to create a user principal. You only need to create a user principal for this host:

```
kadmin: addprinc anna
```

For this principal, you can choose the same password as a user's UNIX password.

Now you're ready to test it. Try to telnet, rlogin or ftp to your client as a Kerberos user and watch how the authentication works.

Because Kerberos uses `gethostbyname()` to determine the fully qualified domain name, make sure that it works correctly on your system. To test this, run:

```
# /krb-5.1.1.1/src/tests/resolve/resolve <hostname>
```

If it doesn't bring you a fully qualified hostname (e.g., oak.tree.com), the KDC will deny authorization. To fix it, add a long name entry into the /etc/hosts file (e.g., 192.168.10.1 oak.tree.com oak). Also make sure that you use kerberized services and utilities. Put the /usr/local/kerberos/bin and the /usr/local/kerberos/sbin first in your PATH.

For more information about Kerberos administration and usage, refer to the *Kerberos V5 System Administrator's Guide* and the *Kerberos V5 UNIX User's Guide* that come with the distribution.

Sun's implementation of Kerberos

Before Solaris 8, Sun used to ship a basic set of Kerberos V4 utilities. The RPC that comes with this also supports Kerberos V4. Sun has also announced a Sun Enterprise Authentication Mechanism (SEAM) that's based on the Kerberos V5 standard (currently only for Solaris 2.6 and 7). This is a component of the Solaris Easy Access Server. Sun developed this product for integration with Microsoft's Windows 2000 networks.

SEAM software supports a Java technology-based administrative tool for easy access and configuration. It also enables users to load authentication information in batch mode, which is particularly useful if your enterprise loses or gains large numbers of users each year. For more information about SEAM, check **www.sun.com/software/solaris/ds/ds-seamss/**.

Limitations

Even though Kerberos is a strong security solution, it also has several limitations. First, Kerberos makes no provisions for host security; it assumes that it's running on trusted hosts with an untrusted network. If your host security is compromised, then Kerberos is compromised as well.

Second, Kerberos stores all passwords encrypted with a single key. This means that in the event the Kerberos Server is compromised, all user passwords must be changed.

Third, Kerberos uses a principal's password (encryption key) as the fundamental proof of identity. If an attacker steals a user's Kerberos password, then the attacker can impersonate that user with impunity.

Finally, Kerberos doesn't work well in a multi-user environment. It keeps tickets in the /tmp directory. It's possible that another user sharing the same workstation can steal the user's ticket. Stolen tickets can then be used to obtain fraudulent service. ✳

Listing E: *Add the following lines to /etc/inet/inetd.conf*

```
klogin   stream  tcp  nowait  root /usr/local/kerberos/sbin/klogind klogind -k -c
eklogin  stream  tcp  nowait  root /usr/local/kerberos/sbin/klogind klogind -k -c -e
kshell   stream  tcp  nowait  root /usr/local/kerberos/sbin/kshd kshd -k -c -A
ftp      stream  tcp  nowait  root /usr/local/kerberos/sbin/ftpd ftpd -a
telnet   stream  tcp  nowait  root /usr/local/kerberos/sbin/telnetd telnetd -a valid
```

SOLARIS Q & A

Using external SCSI devices with Solaris x86 on a PC system

I've just installed Solaris x86 on a PC system, and I want use some external SCSI devices, especially my SCSI Zip drive. Can you suggest the best way to do this?

Zip drives are a popular mass storage device, commonly used by graphic artists and other users with large volume requirements, but usually with some flexibility regarding access. On a Solaris system, Zip disks are useful for backing up system data and individual user files (using a ufsdump script). You can install support for a SCSI Zip disk quite easily by using the instructions on the following Iomega Web sites:

www.iomega.com/support/documents/4019.html

www.iomega.com/support/documents/2019.html

The Iomega approach is to define the Zip drive as a device and enter its data into /etc/format.dat. A

user makes an entry detailing all of the drive's operational parameters, like that shown in Listing A.

Listing A: *Entry in format.dat allows us to see a Zip drive*

```
disk_type="Zip 100"\
                    :ctlr=SCSI\
                    :ncyl=2406:acyl=2:pcyl=2408:nhead=2\
                    :nsect=40:rpm=3600:bpt=20480
      partition="Zip 100"\
                    :disk="Zip 100":ctlr=SCSI\
                    :2=0,192480
                    :2=0,1159168
```

Ziptool

Since Iomega doesn't provide a high level of support for Solaris (especially for disk formatting), there are several freeware programs available that attempt to replicate the functionality of the product running under different operating systems. One such product is Ziptool, which you can download from

fy.chalmers.se/~appro/ziptool.html

Significantly, Ziptool provides support for disk locking and enhanced read/write access control. It also supports disk labeling, formatting, and automated volume management, which is more than you'll get by treating the drive as a SCSI disk directly. Ziptool installs as a set-uid root program, which means that ordinary users who are granted access to the Zip drive could potentially exploit any malicious code in the software, if it existed. However, since many device access provisions were relaxed in Solaris 2.x compared to Solaris 1.x, this level of access would be consistent with most organization's use policies.

Installing Ziptool

To install the program, download it from the Web site mentioned to a temporary directory. Unpack the sources, and compile them using this command:

```
unix% gcc -o ziptool ziptool.c -lvolmgt
```

Since volume management support is provided, it's necessary to link the appropriate library (libvolmgt.a) with your executable. If compilation fails because it can't find this library, check that your LD_LIBRARY_PATH includes the directory for libvolmgt.a.

If compilation is successful, ensure that your Zip drive is plugged into your SCSI port, and powered on. Insert a disk into the drive, and issue the command

```
ziptool dev parameter
```

where `dev` is the raw device name of your drive, and `parameter` is a command string consisting of one of the following:

- `rw`—Temporarily unlocks the volume
- `RW`—Permanently unlocks the volume
- `Ro`—Makes the volume read-only
- `RO`—Enables the volume to be protected by a password
- `eject`—Ejects the current volume
- `noeject`—Prevents the volume from being ejected

You should now be able to use your Zip disks just as effectively with Solaris as you would with any other operating system.

Tracking .rhosts files

I've been reading a lot about .rhosts files lately, and the security exploits associated with its use. I have now created a site policy that states that individual .rhosts files shall not be permitted. How do I go about enforcing this policy?

The .rhosts file is considered dangerous in many situations, unless you're behind a firewall and assume that all local hosts are trusted. This includes dial-up modems and terminal servers which may also be behind the corporate firewall, but through which an intruder could still break into your system. The best way to determine whether there are risks in your current setup is to examine the system logs for suspicious activity, and/or run a penetration-testing tool like SATAN or SAINT, which will identify potential vulnerabilities for you automatically.

Even with strict usage policies, many users will still attempt to use .rhosts because of the convenience of being able to remotely log on between machines without having to type a username or password. Since user .rhosts files aren't directly controlled by the super user, it's difficult to directly police this kind of policy manually, especially for machines with thousands of users.

Fortunately, Solaris provides tools for tracking down and deleting these files. These tools, combined with a cron job for automatically executing a deletion job, provide a safe and easy way to eliminate any potential threats. The `find` command can be set up to search all disks, or only partitions

that you specify, and execute a shell command (such as rm). Thus, find can locate all instances of .rhosts and delete them. A cron entry for a root like

```
30 4,10,16,22 * * * find /home -name
➥.rhosts -print -exec rm{} \;
```

locates every instance of .rhosts on the filesystem for local users, and deletes the offending file every six hours. ✳

The Solaris Q & A column is written by Paul A. Watters. If you have a question about Solaris, send it to pwatters@mpce.mq.edu.au.

Solaris CD recording

by Clayton E. Crooks II

As CD-Recordable (CD-R) and CD-Rewritable (CD-RW) drive prices have continued to plummet over the last year, they have become an even better choice for data archiving and distribution. Although industry standard tape is an effective backup mechanism, the number of tape sizes and format incompatibilities prevent it from being used as an effective tool for data distribution. On the other hand, CD-Rs can be read on any UNIX operating system running on any platform. Depending on your experience and the amount of money you wish to spend, there are several options now available for recording CDs under Solaris.

Hardware

To successfully record either audio or data CDs, you'll need to purchase a CD-R or CD-RW drive. Although hundreds of drives are currently on the market, only a select few will work with Solaris. These drives are available from several companies, and many are exactly the same drives with different exterior components or labels to disguise their origins. Plextor, Ricoh, and Smart & Friendly make drives that are reported to work most reliably under Solaris. Note that it's the mastering software that will dictate your drive choices. For SCSI drives, you can attach the drive as SCSI device ID 6 (the standard SCSI ID for a CD-ROM drive on a Sun workstation) and it will also function as a CD-ROM drive.

CD-R or CD-RW?

The difference between a CD-RW and a CD-R is that you can erase and rewrite to CD-RW discs. While CD-R is Write-Once, Read-Many (WORM), CD-RW gives read and write, and overwrite (erase) access to the disc. Other than that, CD-R and CD-RW discs are essentially the same. However, because you have the ability to erase discs with CD-RW, the software may handle them in a slightly different way.

CD-R/CD-RW vs. CD-ROM

CD-R and CD-RW media aren't the same as CD-ROM media. A laser must write the contents of the CD-R and CD-RW; a CD-ROM is manufactured using injection molding with a thin layer of aluminum supplied by a coating process called sputtering. It's this thin aluminum layer that is read by the CD-ROM drive.

Because the CD-RW reflectivity is lower than a comparable CD-ROM disc, the media may not be readable by every CD-ROM drive. Most new CD-ROM drives do support CD-RW media, but most of them cannot read CD-RW discs at full speed. A few older audio CD players and many new ones can handle CD-RW discs, but many cannot. If you want to create audio CDs on CD-RW media, make sure that your player can handle them.

Packet writing

The software you use for CD-R and CD-RW is basically the same, and you can use several approaches for writing data to both. Packet writing is one such option; it allows several writes for each individual track and uses only seven blocks of overhead per write. Since you can write packets that are small enough to fit entirely in the CD recorder's buffer, the risk of buffer underruns is reduced or eliminated.

Unfortunately, there are some problems with packet writing. The biggest problem is that many older CD-ROM drives can't deal with the gaps between packets. In fact, they frequently become baffled if they read into the gap—a problem that is exacerbated by read-ahead optimizations on some models.

CD-R and CD-RW can also use disc-at-once (DAO) and track-at-once (TAO) recording. DAO writes the entire CD in one pass, possibly writing multiple tracks. The entire burn completes without interruption, and no further information can be added at any time.

TAO allows you to do the writes in multiple passes. There's a minimum track length of 300 blocks (which equals approximately 600 KB for data CDs). It can't consume more than a maximum of 99 tracks per disc. Another disadvantage to TAO is the additional overhead associated with stopping and restarting the laser. Because the laser is turned on and off for each track, the recorder leaves a couple of blocks between tracks, which are called run-in and run-out blocks. If done correctly, the blocks will be silent and usually indistinguishable. However, some combinations of software and hardware may leave garbage in the gap, which can result in a slight, but annoying, click between tracks.

CD-RW media is generally more expensive than CD-R, but recent price reductions have narrowed the gap considerably. There's a limit to the number of times an area of the disc can be rewritten, but that number is relatively high, and some manufacturers have claimed as many as 100,000 rewrites without failure.

The recording process

The recording of the media is nearly identical. CD-RW drives use phase-change technology, which creates deformations in the recording dye layer. The state of material in the recording layer changes from a crystalline to an amorphous form. These states have different refractive indices and can be optically distinguished. On the other hand, a laser in the CD-R creates a series of holes in the disc's dye layer called *pits*, and the spaces between the pits are called *lands*. The pattern of pits and lands on the disc encodes the information and allows it to be read by a CD-ROM or audio CD drive.

Mastering software

As we mentioned earlier, keep in mind that the software is usually the factor that determines if you can use a particular make and model of a recorder. Be sure to check the literature for your respective software.

cdrecord

The first, and lowest cost, option is a freeware application entitled *cdrecord*. As this software is freeware, the available technical support is limited. However, it's good software that deserves a look.

You can run cdrecord without problems on a Solaris system and use the SVr4 priocntl() call to establish SVr4 real-time scheduling. This call

About our contributors

Clayton E. Crooks II is a self-employed computer consultant living in Knoxville, TN. He's married with one child. His hobbies include game development, 3-D modeling and any athletic activity he can find time for.

Boris Loza holds a Ph.D. in computer science. He worked as a UNIX administrator and developer for 10 years. Currently he's working for Fidelity Investments Canada in the position of Data Security and Capacity Planner. He has a daughter Anna and likes reading computer and mystery books, and watching movies. You can reach him at Boris.Loza@FMR.com.

Paul A. Watters is the project manager at Neuroflex, where he's responsible for developing natural language database systems in Java on the Solaris platform. He can be reached at pwatters@ mpce.mq.edu.au.

Inside Solaris™
Tips & Techniques for users of Sun Solaris

Inside Solaris (ISSN 1081-3314) is published monthly by Element K Journals, a division of Element K Press, 500 Canal View Boulevard, Rochester, NY 14624.

Customer Relations

US toll free .....(800) 223-8720
Outside of the US.....(716) 240-7301
Customer Relations fax .....(716) 214-2386

For subscriptions, fulfillment questions, and requests for group subscriptions, address your letters to

Element K Journals Customer Relations
500 Canal View Boulevard
Rochester, NY 14623

Or contact Customer Relations via Internet email at journals@zdu.com.

Editorial

Editor.....Garrett Suhm

Assistant Editor .....Jill Suhm
Managing Editor .....Michelle Rogers
Copy Editors.....Rachel Krayer
Glenna Lechner
Contributing Editors .....Clayton E. Crooks II
Boris Loza
Paul A. Watters
Print Designer .....Melissa Ribaudo
Cover and Content Design.....Melissa Ribaudo

You may address tips, special requests, and other correspondence to

The Editor, *Inside Solaris*
500 Canal View Boulevard
Rochester, NY 14623

Editorial Department fax .....(716) 272-0064

Or contact us via Internet email at inside_solaris@elementkjournals.com.

Sorry, but due to the volume of mail we receive, we can't always promise a reply, although we do read every letter.

Element K Journals

General Manager..... Kelly Baptiste
Manager of Customer Relations ..... Heather Loiacono
Manager of Operations ..... Cristal Haygood
Manager of Print Design ..... Charles V. Buechel
Manager of Product Marketing ..... Mike Mayfield
Senior Product Marketing Manager ..... Brian Cardona

Postmaster

Periodicals postage paid in Rochester, NY and additional mailing offices.

Postmaster: Send address changes to

Inside Solaris
P.O. Box 92880
Rochester, NY 14692

Printed in the USA.

Price

Domestic .....$129/yr ($11.00 each)
Outside US .....$149/yr ($13.00 each)

Our Canadian GST# is: R140496720. CPM# is: 1446703.
QST# is: 1018491237.

Back Issues

To order a back issue from the last six months, call Customer Relations at (800) 223-8720. Back issues cost $11.00 each, $13.00 outside the US. You can pay with MasterCard, VISA, Discover, or American Express.

Are you moving?

If you've moved recently or you're planning to move, you can guarantee uninterrupted service on your subscription by calling us at (800) 223-8720 and giving us your new address. Or you can fax us your label with the appropriate changes at (716) 214-2386. Our Customer Relations department is also available via email at journals@zdu.com.

Coming up...

- Proxy caching with Squid
- BIND conversions

runs cdrecord in a higher priority than all kernel processes. On systems that provide POSIX real-time scheduling, your results may be slightly worse, as POSIX RT doesn't seem to grant the latter real-time behavior.

The cdrecord distribution contains a SCSI user level transport library that's suitable to talk to any SCSI device without having a special driver for it. You'll need the SCSI general driver scg, which was developed by Jörg Schilling, in order to run cdrecord.

The distribution for Solaris, available from **www.fokus.gmd.de/research/cc/glone/employees/joerg.schilling/private/cdrecord.html**, needs to be compiled in order to work. According to the documentation, you should never compile anything in a locale other than C, unless you're sure that your C-compiler and the program nm have been patched. However, with Solaris 2.6 or later, and with the Workshop 5.0 compiler or later, this isn't needed.

To ensure that local C is set, run the following commands before running make:

```
Setenv LC_ALL C
in csh
or
LC_ALL=C
export LC_ALL
in sh
```

This step is necessary because Solaris nm, before 2.5.1, dumps core in any other location. For that reason lorder won't work. Unfortunately, a patch isn't available, but Sun fixed the problem with Solaris 2.5.1. If you want to perform compilations on Solaris, it's best not to have /usr/ucbin in your PATH. If you want to have /usr/ucb in the PATH, it must be the last entry. To be able to use `make` and `ld`, you need to make sure that /usr/ccs/bin is in your path.

One final note before using the software: you must disable volume management before creating a CD. You need to do this because newer drives identify themselves correctly as CD-ROM drives. Unfortunately, the volume management daemon from Sun doesn't check that there may be a recordable medium in the drive. To do this permanently on Solaris, edit /etc/vold.

If you're planning on using cdrecord for your mastering software, we recommend downloading and installing BurnIt from **sunsite.auc.dk/BurnIT/**. It's a Java front end to cdrecord that enhances the usability of the software.

Gear

Gear Professional for Solaris is another option to consider. It offers all of the regular benefits of mastering software, along with additional features like data integrity. These features include:

- Post-gap writing on all recorders
- Write-once data protection
- Universal multi-session incremental writing
- Full error code handling and device testing
- Full verification of formatted data

Gear allows you to master virtually any CD-ROM format, including DA, XA, XA interleaving, CD-enhanced, and Mixed Mode. It includes support of the universal standard RockRidge, which enables users to read files off a written CD on any other UNIX system exactly as they appeared on the original workstation—including permissions and passwords.

Gear includes unique scripting functions that allow for unattended recording, multiple copy creation, and project backup, and may be the most desirable of all the Gear benefits. These advanced features, along with support for multi-platter systems, are the options that really set Gear apart.

Final thoughts

Although several recording solutions are available for Solaris, Gear has the best feature set and longest history. On the other hand, cdrecord doesn't cost anything, and is more than adequate for most jobs. As both are quite capable, the support, or lack of, may be the determining factor in your decision. Once you have decided on the mastering software, you should check their documentation to determine the available recorders. ✳